iTAP: In-network Traffic Analysis Prevention using Software-Defined Networks

https://itap.ethz.ch

Roland Meier
ETH Zürich
meierrol@ethz.ch

David Gugelmann
ETH Zürich
gugelmann@tik.ee.ethz.ch

Laurent Vanbever
ETH Zürich
lvanbever@ethz.ch

ABSTRACT

Advances in layer 2 networking technologies have fostered the deployment of large, geographically distributed LANs. Due to their large diameter, such LANs provide many vantage points for wiretapping. As an example, Google's internal network was reportedly tapped by governmental agencies, forcing the Web giant to encrypt its internal traffic. While using encryption certainly helps, eavesdroppers can still access traffic metadata which often reveals sensitive information, such as who communicates with whom and which are the critical hubs in the infrastructure.

This paper presents iTAP, a system for providing strong anonymity guarantees *within* a network. iTAP is network-based and can be partially deployed. Akin to onion routing, iTAP rewrites packet headers at the network edges by leveraging SDN devices. As large LANs can see millions of flows, the key challenge is to rewrite headers in a way that guarantees strong anonymity while, at the same time, scaling the control-plane (number of events) and the data-plane (number of flow rules). iTAP addresses these challenges by adopting a hybrid rewriting scheme. Specifically, iTAP scales by reusing rewriting rules across distinct flows and by distributing them on multiple switches. As reusing headers leaks information, iTAP monitors this leakage and adapts the rewriting rules before any eavesdropper could provably de-anonymize any host.

We implemented iTAP and evaluated it using real network traffic traces. We show that iTAP works in practice, on existing hardware, and that deploying few SDN switches is enough to protect a large share of the network traffic.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SOSR '17, April 03-04, 2017, Santa Clara, CA, USA
© 2017 ACM. ISBN 978-1-4503-4947-5/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3050220.3050232

CCS Concepts

•Security and privacy → Pseudonymity, anonymity and untraceability; *Network security;* **•Networks → Network privacy and anonymity; Programmable networks;**

Keywords

Anonymous communication; wiretapping; SDN

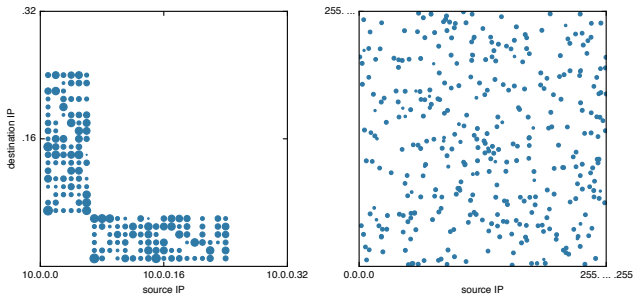
1. INTRODUCTION

Since the Snowden revelations, it is well-known that network eavesdropping was (and probably still is) performed in the Internet core, particularly on undersea cables [22]. While worrying, these threats can be mitigated to a large degree by hiding connection metadata (e.g., using Tor [14]) and by relying on pervasive encryption (e.g., using VPNs).

However, network eavesdropping is not limited to the Internet backbone. As enterprise networks become bigger in terms of users and physical reach, they, too, become susceptible to wiretapping. Indeed the advent of new layer 2 technologies such as TRILL [7], Shortest Path Bridging [30], or SDN-based solutions [34] enables network administrators to build large LAN zones that can easily span several thousand devices and users. Due to their large physical diameter, such networks inevitably exhibit many vantage points for wiretapping. Actually, the majority of the attacks is now performed by insiders, *i.e.* malicious insiders or inadvertent actors [4] that act from within the network, rather than by remote attackers. As an example, Google's Wide-Area Network (WAN) – the private network connecting Google's data centers – was reportedly tapped by governmental agencies, forcing the Web giant to encrypt its internal traffic [36].

While encrypting internal traffic protects the payload of connections, an in-network attacker can still monitor and analyze the unencrypted packet headers, i.e., the *metadata*. In particular, the MAC addresses and, in case of SSL/TLS application layer encryption, also the IP addresses of the source and destination hosts along with the source and destination ports used for the communication.

By analyzing these unencrypted header fields, an attacker can gain useful information about: *i)* which hosts communicate; *ii)* the topology of the network; *iii)* the addresses of



(a) Regular network (b) iTAP-enabled network

Figure 1: Example of information leakage. Observed source and destination IP addresses without (a) and with iTAP (b).

important hosts driving a lot of traffic (*e.g.*, data storage) or a lot of flows (*e.g.*, authentication server). An eavesdropper can use the collected metadata to extract privacy-relevant information, or to identify valuable targets during the reconnaissance phase of an attack. To give some examples: *First*, an attacker, engaged by a competitor, could easily determine whether a certain person (*e.g.*, the CEO) is currently working or on holidays by monitoring network traffic. Her absence might indeed be a good occasion to launch a new product or a campaign against the attacked company. *Second*, an attacker can foresee the future actions of the company [41]. Indeed, the launch of a new product is often preluded by a significantly higher-level of communication between a broad range of departments. *Third*, an attacker can infer the addresses of important or heavily-used hosts (*e.g.*, authentication and accounting servers) as well as who is accredited to use them. In a following step, the attacker could then run a targeted attack against these clients or servers.

This work. We present iTAP, a partially deployable network-level system that enables anonymous communication *within* the premises of a network. The key insight is to leverage the programmability offered by Software-Defined Networking (SDN) to rewrite packet headers at the network edge to varying randomized identifiers. iTAP is useful with as few as two SDN devices and its anonymity guarantees grow linearly with the number of SDN devices from thereafter.

With iTAP in place, in-network eavesdroppers only see randomized IDs (which carry no information) as headers and cannot single out hosts unless they eavesdrop on all links of a path, which is unlikely in practice. As an illustration, Fig. 1 shows the distribution of source and destination IP addresses observed by a link-level eavesdropper with and without iTAP. Without iTAP, it is evident that 18 clients and six servers are communicating with each other. With iTAP, the observed IP addresses are spread over the whole address range and neither the real addresses nor the number of communicating hosts is recognizable.

Challenge: Providing network anonymity at scale. Rewriting packet headers for thousands of hosts and millions of flows while ensuring strong anonymity guarantees is challenging. From an anonymity viewpoint, the best rewriting solution consists in assigning one random ID per TCP/UDP flow. Yet, not only does this require millions of forwarding rules in the data plane, but it also mandates a purely reactive controller which has to be involved in each connection establishment. From a scalability viewpoint, the best solution would be to assign one random ID per host and reuse it for all flows pertaining to that host. Doing so would only require two forwarding rules per host at the edge, i.e., a marginal increase with respect to today's networks, and could even be done proactively so that the controller does not have to be involved in any connection establishment. However, this would not provide strong anonymity, e.g., an attacker could still easily count the number of devices in the network.

iTAP solves the scalability and privacy problems of these approaches while retaining their strengths by employing a hybrid approach. In particular, iTAP reuses IDs across distinct flows so as to maintain the overall number of forwarding rules to an acceptable level. Reusing rules leaks some information out to potential eavesdropper, namely which bits in the IDs identify hosts and where these hosts are connected. iTAP addresses this problem by continuously monitoring the amount of leaked information and by adapting the rewriting scheme *before* any attacker could provably (relying on information theory and the notion of the unicity distance) learn enough information to break the used scheme.

Reusing IDs enables iTAP to maintain $\mathcal{O}(\#\text{hosts})$ forwarding rules in the core and $\mathcal{O}\left(\#\text{hosts}^2\right)$ at the edge. While this number of forwarding rules is perfectly manageable for users-facing edge switches (see §9), it can however be a problem for Internet-facing edges which act as focus points for any flow directed to the outside. iTAP addresses this problem by monitoring each edge switch (in terms of flow rules or updates per second) and by offloading obfuscation rules for Internet destinations on other switches before any switch limit is reached. While offloading obfuscation rules reveals the actual Internet destinations on some links, the anonymity of internal hosts is still protected anywhere in the network and an attacker cannot know who is communicating with a particular destination.

Practical results. iTAP works in practice. We implemented a prototype on top of Floodlight [43] and evaluated iTAP using real traffic traces. Our results show that iTAP can handle the load of a large network using commodity SDN hardware.

Novelty. Communicating anonymously has been the focus of extensive research [12–14, 24]. Yet, we are not aware of any other work which focuses on anonymizing *internal* communications without requiring host support. In particular, Tor [14] and Hornet [13] both require host support. Recently, MACSec [5] has been developed to provide link-based encryption for traffic beyond L2 (MAC addresses are not encrypted). In contrast, iTAP provides network-based anonymization for traffic beyond L1 (i.e., the MAC addresses are also protected). Furthermore, iTAP provides anonymity guarantees when partially deployed, which is not the case for MACSec.

Main contributions. Our main contributions are:

- The design of iTAP (§3), a network-level anonymity system which provides strong anonymity guarantees in the presence of a powerful attacker (§2). With iTAP, an eavesdropper cannot determine whether two flows originate or terminate at the same host (assuming payload encryption).
- A header rewriting scheme (§4) which enables iTAP to scale to large networks while providing strong anonymity guarantees. As an added benefit, our rewriting scheme enables iTAP to locate active attackers (§6).
- An efficient migration strategy for iTAP which allows to gradually increase the anonymity level in a network (§7).
- A prototype implementation (§8) of iTAP on top of the Floodlight SDN controller. We will make our implementation publicly available.
- A comprehensive evaluation based on real traffic traces containing millions of flows representing the activity of 400 users and showing that iTAP can handle realistic network load while preserving user anonymity (§9).

2. NETWORK AND THREAT MODEL

In this section, we describe the network iTAP is designed to be used in and the threat iTAP is designed to protect from.

Network model. We consider the case of an internal network which is managed by a single managing authority. We assume that the network is composed of traditional devices, *i.e.*, routers and switches, along with a subset of SDN switches. Routers are running normal routing protocols (*e.g.*, OSPF and BGP) while traditional switches are running some flavors of the Spanning Tree Protocol (STP). In the rest of the paper (and without loss of generality), we assume that the network is mostly based on L2 technologies, with most of the routing being done at the gateway to the Internet—one of the most common design patterns in enterprise networks [3]. Fig. 2 illustrates an example of such a network.

SDN switches are controlled by one or more (for redundancy) iTAP controller via secure (TLS-enabled) connections. The iTAP controller also participates in any routing or spanning tree protocol to learn about the legacy topology and to interact with it. By programming forwarding rules in the SDN switches, iTAP can arbitrarily rewrite the packet headers as well as influence the forwarding of packets crossing these devices. We refer to the first SDN-switch that a packet traverses as *ingress* switch. Similarly, we refer to the last SDN switch traversed by a packet before reaching the destination as *egress* switch.

Since iTAP focuses on anonymizing packet headers, we assume that the payload of the transport layer protocol is either encrypted or does not reveal information about the communicating parties.

Attacker model. We consider an active (or passive) network attacker whose goal is to collect useful information about the network including: *(i)* who is communicating with whom; *(ii)* the presence or absence of particular users; *(iii)* the number of flows initiated or received by each host; *(iv)* the topology; or *(v)* the addresses of potential attack-targets.

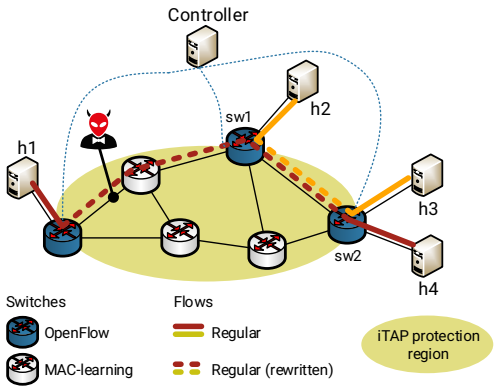


Figure 2: iTAP overview. The network consists of SDN-enabled switches and traditional L2-switches. The flows are rewritten between the ingress and the egress switch.

For doing so, the attacker can: *(i)* eavesdrop on a subset of the internal network links and have full access to the exchanged traffic (Ethernet headers and payloads); *(ii)* arbitrarily inject packets on eavesdropped links; and *(iii)* control a set of hosts connected to the network and craft arbitrary network packets on these hosts.

Out of scope. We note that:

(i) The first and last hop, i.e., the network link connecting hosts to the corresponding SDN switches, cannot be protected without installing software on the end host (see §3.3).

(ii) While the attacker can compromise a subset of hosts, we assume that the attacker can neither access nor manipulate the software and configuration running on the networking devices and on the controller. Indeed, while it is relatively easy to tap on links spanning a large geographical region, owning the network devices themselves is much harder. In §10, we relax this assumption and discuss the consequences of having attackers additionally controlling a subset of the SDN switches.

3. OVERVIEW

In this section, we provide an informal overview of our anonymization approach. We first discuss the key mechanisms behind iTAP (§3.1). We then present the core anonymity properties provided by iTAP pertaining to communication, volume and topology, respectively (§3.2). Finally, we describe iTAP's two main limitations (§3.3): information leakage coming from the payload and timing attacks, which is a fundamental limitation of low-delay anonymity systems.

3.1 iTAP

An iTAP-enabled network (see Fig. 2) consists of SDN-enabled switches along with traditional L2 switches controlled by an iTAP controller. The main functionality provided by iTAP is flow obfuscation at scale. To do so, iTAP leverages an *adaptative hybrid obfuscation scheme* where

	no rewriting	*unique ID*	*virtual addresses*	***iTAP hybrid approach***
forwarding paradigm	destination MAC	flow	destination address	**destination ID**
# rules in core switches	$\mathcal{O}$ (#hosts)	$\mathcal{O}$ (#flows)	$\mathcal{O}$ (#hosts)	$\mathcal{O}$ (**#hosts**)
# rules in edge switches	$\mathcal{O}$ (#hosts)	$\mathcal{O}$ (#flows)	$\mathcal{O}$ (#hosts)	$\mathcal{O}$ (**#hosts**^{2})
anonymity guarantees	none	high	medium	**high**

Table 1: Intuitive rewriting techniques are either costly in terms of flow rules or provide little anonymity. In contrast, iTAP hybrid approach provides *both* scalability and anonymity.

multiple flows share the same obfuscation header and where the obfuscation scheme is adapted before any attacker can learn enough information to break it. In addition to traffic obfuscation, iTAP can *detect the attackers position* by monitoring unexpected entry points for obfuscated traffic (e.g., an attacker trying to probe an obfuscated header). iTAP *supports partial deployment* and does not require a network solely consisting of OpenFlow switches to be useful. We now briefly describe these aspects.

Obfuscating traffic at scale (§4). Upon the first flow between two hosts (say h1 and h4 in Fig. 2), iTAP: *(i)* computes an obfuscated header for the pair (h1, h4); *(ii)* installs rewriting rules on the ingress (first) and egress (last) switches along with forwarding rules (matching on a random subset of the obfuscated header) on the core switches; and *(iii)* pushes the first packet of the flow directly to its egress switch. The installed rules can be used for any flow between h1 and h4.

To prevent an attacker from determining the real source and destination of a packet, iTAP obfuscates the addresses, both at L2 (MAC addresses) and at L3 (IP addresses). We refer to the concatenation of all address fields (MAC and IP) as the header bitstring h.

From a pure anonymity viewpoint, the best solution would be to replace h by a random value for each flow. The attacker in Fig. 2 would then observe uniform random addresses. While the rewritten header $\tilde{h}$ would not reveal any information about h, this approach does not scale as the controller would need to be involved for each flow. Switches would also have to carry potentially millions of forwarding rules.

To achieve better scalability, one could replace each addresses individually, that is, the source addresses by a source ID and the destination addresses by a destination ID. All packets from the same source or towards the same destination would then have the same source or destination IDs and the switches forward packets based on their destination IDs. While this approach scales well, it provides poor anonymity guarantees because an eavesdropper can immediately determine which flows have the same source or destination. Considering Fig. 2, the attacker could easily group the flows sharing the same endpoints.

In contrast, iTAP uses a hybrid rewriting approach (see Table 1). In iTAP, $\tilde{h}$ consists of a few bits that identify the source and the destination of the flow and many random bits. As we will describe later, the non-random bits and the random bits are mixed such that for a given $\tilde{h}$, it is impossible to distinguish between non- random and random bits. By using the existing header fields to encode the source and destination identifier, iTAP does not need to attach additional labels to a packet and therefore does neither increase a packet's size, nor decrease the MTU of the network. Considering Fig. 2, the attacker can neither determine the real source or destination of a flow nor can she decide whether two flows originate or terminate at the same host.

Guaranteeing obfuscation quality over time (§5). iTAP continuously monitors the amount of information that is exposed to a potential eavesdropper. As the packet header is the same for each flow between a pair of hosts, an eavesdropper can indeed try to infer which bits are used for forwarding and with this, infer the location of the source and destination hosts. To avoid this, iTAP continuously monitors how much information any attacker could infer and changes the encoding before the attacker could *provably* have learned enough to break iTAP's obfuscation scheme.

Detecting the exact location of attackers (§6). As iTAP is aware of the legitimate flows in the network (along with their location), it can raise an alert if it detects a packet that is not part of a known flow. This alert together with the location where the unexpected packet arrived allows the network operator to locate an attacker that injects traffic at a link. For example, an attacker that injects a packet between two switches (Fig. 2) will be immediately detected unless the injected packets have the same source and destination ID as one of the legitimate flows crossing this link.

Supporting partial deployment (§7). iTAP does not require a network solely consisting of OpenFlow switches. Instead, traditional L2 switches can be used at any location in the network. The controller makes sure that packets which are crossing such legacy switches will have the destination MAC address set to one that belongs to the next OpenFlow switch. Doing so, iTAP can protect a segment of the network with as little as two OpenFlow switches (e.g., sw1 and sw2 in Fig. 2) or the *whole* network with (software or hardware) OpenFlow switches at the edge.

3.2 Anonymity guarantees

iTAP provides three core privacy guarantees for packets within iTAP's protection region, i.e. the links between the ingress and egress SDN switches:

[G1] Communication anonymity. iTAP hides who is communicating with whom. By replacing the source and destination addresses by pseudo-random identifiers, iTAP makes it impossible for eavesdropper to determine the real source and destination based on the packet header. While iTAP does

not hide the TCP/UDP ports by default, it could easily do so as well. As described above, this comes at the price of scalability if performed on all flows. Yet, it could be done for specific applications (i.e., specific ports).

[G2] Volume anonymity. iTAP hides how much traffic (measured in bytes or number of connections) any two hosts exchange. Specifically, iTAP's adaptive obfuscation scheme prevents an eavesdropper from inferring that a given set of flows are exchanged between the same endpoints. Among others, this prevents the eavesdropper from identifying important hosts (acting as potential choke points).

[G3] Topology anonymity. iTAP hides the number of hosts in a network. As an eavesdropper can not identify the network host identifiers in packet headers, she can not count the number of active network hosts.

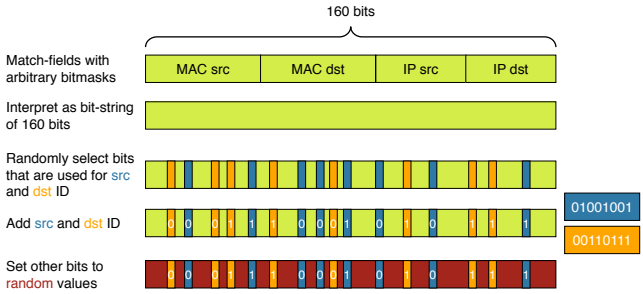


Figure 3: Procedure to rewrite a flow header. The OpenFlow match fields supporting arbitrary bit-masks are concatenated and interpreted as a header bit-string of length 160 bits. The source ID and the destination ID is encoded at randomly chosen bit positions. Unused bits are set to random values.

3.3 Limitations

As an anonymity system, iTAP focuses on protecting the header (i.e., the metadata) of the communication, *not* the communication content which is assumed to be encrypted. Similarly to other low-delay anonymity system such as Tor [14], iTAP is sensitive to timing correlation attacks (e.g., [35, 37]) – even though it makes them harder. Finally, iTAP does not protect the first hop of the network as it resides solely in the network, not on the host. We discuss these limitations in the following.

[L1] iTAP does not protect against information leakage from the payload. iTAP does not modify the payload and can therefore not prevent information leakage there. For example, if the used encryption scheme reveals information about the communicating hosts or there is no encryption used, an eavesdropper might be able to extract sensitive information despite the protection provided by iTAP. Besides applying suitable encryption on the payload, iTAP could be extended to work with P4 [10], a high-level language for packet-processors that provides even more flexibility than OpenFlow (regarding the modification of packet contents on the switch, for example). Or, as an additional alternative, iTAP can tunnel packets that reveal sensitive information (for example the certificate exchange in SSL/TLS) through the controller.

[L2] iTAP does not protect against payload and timing correlation attacks. iTAP achieves anonymous communication by rewriting addresses in the packet header. It does neither modify the payload nor does it change the order of packets and can therefore not protect against payload or timing analysis attacks. However, iTAP makes payload or timing correlation attacks against particular hosts harder because an attacker can no longer rely on packet headers to identify packets related to the host. That is, before being able to mount a corresponding attack, the attacker needs to identify packets belonging to the hosts of interest by means of expensive payload analysis.

[L3] iTAP does not protect against an eavesdropper sitting on the first hop or on the host directly. Being network-based, iTAP's protection region starts (resp. ends) at the first (last) switch encountered, it therefore does not protect from an eavesdropper sitting on the last hop or who has compromised a host. This is essentially a design choice. By making iTAP a pure *in-network* anonymity system, it efficiently protects any internal communication, from any connected devices. If privacy on the first/last link is a primary concern, one can install a software switch [44] on the end host, similar to a VPN client. This provides some privacy guarantees on the edge link too, as we will discuss in §10.

[L4] iTAP does not hide the overall amount of traffic in a network While iTAP hides the amount of traffic that any two hosts exchange, it does not change the amount of traffic, nor the number of flows, which can then still be determined by an eavesdropper.

4. REWRITING PACKET HEADERS

We now describe the two techniques iTAP uses to provide strong anonymity properties at scale: *(i)* a hybrid obfuscation scheme (§4.1); and *(ii)* the ability to distribute obfuscation rules on multiple switches (§4.2).

4.1 Hybrid obfuscation scheme

As explained in §3, the best obfuscation scheme consists in allocating a pseudo-random identifier to each flow. With this scheme, the only information an eavesdropper sitting on a link could infer from the headers is the number of flows (not hosts) traversing the link. While this scheme provides communication, volume and topology anonymity, it does not scale, be it in the control plane (it requires the controller and all on-path switches to be involved upon each new flow) nor in the data plane (it requires all switches to maintain per-flow state). At the other end of the spectrum, allocating a pseudo-random identifier to each host, not flow, would scale well but would not provide any of the anonymity guarantees.

In practice, we consider that $|h| = 160$ bits corresponding to the MAC (source and destination) and IP (source and destination) fields. Observe that hardware switches compatible with OpenFlow 1.3 support arbitrary bit-masks match

on these fields, enabling flow rules to flexibly match on an embedded $\tilde{h}$, no matter its position in the header.

iTAP uses a hybrid obfuscation scheme which combines the anonymity benefits of a flow-based obfuscation with the scalability properties of per-host obfuscation. Specifically, iTAP relies on relatively small pseudo-random host-based IDs $\tilde{h}$ which are randomly mixed inside a larger randomized packet header h (i.e., $|\tilde{h}| \ll |h|$). This mixing prevents an eavesdropper from knowing which bits belong to $\tilde{h}$ (which would not be the case if $|\tilde{h}| = |h|$). However, unlike per-flow obfuscation, this property does not hold over time. The more flows an attacker sees, the more she can figure out which parts of the header are fixed and infer the corresponding $\tilde{h}$. We describe in §5 how iTAP monitors this leakage of information by preventively adapting $\tilde{h}$ before the attacker could provably have seen enough flows.

Obfuscation procedure. To encode a source ID and a destination ID into a flow, the controller takes the following steps:

1. If the destination is not known yet or the obfuscation mask corresponding to the destination can not be used anymore (§5), the controller creates a new mask and chooses a uniformly random destination ID. Otherwise, the controller uses the existing mask and destination ID.
2. If the mapping from the source host to a source ID does not yet exist for the given destination, the controller chooses a uniform random source ID. iTAP can assign the same host multiple source or destination IDs to increase the entropy in packet headers.
3. The controller composes the rewritten header $\tilde{h}$, which is a uniform random bitstring combined with the bits of the source and destination IDs at the locations defined by the obfuscation mask (see Fig. 3).

4.2 Managing flow rules in the data plane

We now explain how iTAP manages data plane resources by distributing flow rules onto multiple devices, if needed.

iTAP maintains two types of flow rules in the data-plane:

1. **Rewriting rules** perform the mapping from the real to the rewritten header: $\mathcal{M} : h \mapsto \tilde{h}$ and vice-versa;
2. **Forwarding rules** forward packets based on their destination ID (by matching on bits pertaining to $\tilde{h}$). Before traversing a legacy switch, the forwarding rule also rewrites the destination MAC address to one belonging to the next OpenFlow-switch (§7).

To maximize anonymity, rewriting rules should be located as close as possible to hosts, ideally at the edge switch, with only forwarding rules located in the network core. iTAP therefore requires $\mathcal{O}\left(\#\text{hosts}^2\right)$ rewriting rules at the edge and $\mathcal{O}\left(\#\text{hosts}\right)$ rules in the core. While these requirements readily enable iTAP to deal with large networks (§9), some edge switches might not be able to cope with the $\mathcal{O}\left(\#\text{hosts}^2\right)$ rewriting rules, especially Internet-facing ones.

Distributing rewriting rules. iTAP manages to keep the number of flow rules within the hardware limits by distributing the rewriting rules on multiple switches, if needed.

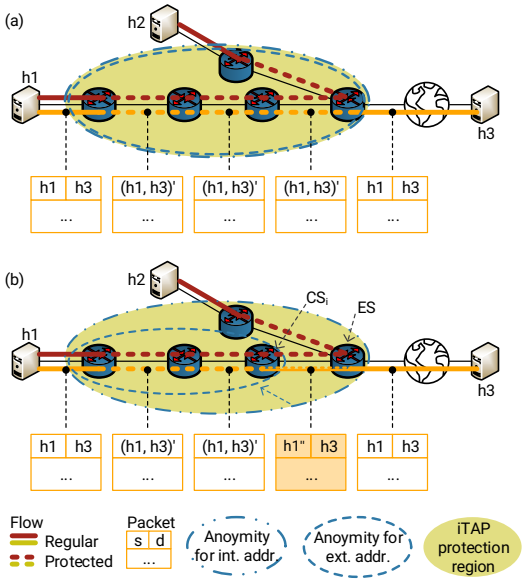


Figure 4: As long as the resource constraints allow it, iTAP fully protects every flow within the whole network *(a)*. Before the flow table of an edge switch congests, iTAP delegates part of the rewriting to a core switch, which reverts the rewriting of the external destination address *(b)*. As a consequence, h3 appears in clear in the highlighted packet.

Thereby, iTAP reduces the number of required rules at an edge switch from $\mathcal{O}\left(\#\text{hosts}^2\right)$ to $\mathcal{O}\left(\#\text{hosts}\right)$. Intuitively, iTAP can offload edge switches by rewriting headers earlier (resp. later) on in the network. Observe that shifting rewriting rules can be done *without interrupting the affected flows*.

Technically, shifting the rewriting of an external address `dst` from the edge switch `ES` by one hop to the core switch $\texttt{CS}_i$ works as follows. First, iTAP installs de-obfuscation rules for all source addresses that are currently communicating with `dst` at `ES`. In a first phase, these rules will not match any packets because the arriving packets towards `dst` still have a completely obfuscated header. Next, iTAP installs rules in $\texttt{CS}_i$ to map the completely obfuscated header to one where only the source address is obfuscated. As soon as these rules are active, `ES` starts receiving packets towards `dst` where only the source address is obfuscated. These packets match the previously installed rules and the old rules for the completely obfuscated header will automatically be removed after an idle timeout.

Moving rewriting rules inside the network also means that an eavesdropper could now observe unobfuscated addresses on some links. To limit the anonymity loss, iTAP only offloads rewrite rules for external (i.e., Internet) addresses, *never* for internal ones.

We illustrate iTAP's distribution process with an example (see Fig. 4). As the load of the Internet-facing switch `ES` approaches its limits (given by its specification), iTAP delegates parts of the rewriting rules to a core switch $\texttt{CS}_i$. For

a flow between an internal host and an external (Internet) server, $\texttt{CS}_i$ de-obfuscates the destination address for outgoing traffic, which leads to packets with an un-obfuscated destination address and an obfuscated source address being sent from $\texttt{CS}_i$ to ES. As a consequence, ES only needs to de-obfuscate the source address (requiring $\mathcal{O}(\#\text{hosts})$ rules). For returning packets, ES obfuscates the (internal) destination address ($\mathcal{O}(\#\text{hosts})$ rules) and sends the packets to $\texttt{CS}_i$ which performs the complete obfuscation of source and destination addresses ($\mathcal{O}\left(\#\text{hosts}^2\right)$ rules).

As the packets for offloaded rules contain un-obfuscated addresses before they leave the network, an eavesdropper located on the link between $\texttt{CS}_i$ and ES can see the real destination (source) addresses for outgoing (incoming) traffic. However, note that: *(i)* internal addresses are always obfuscated; and *(ii)* the eavesdropper cannot determine whether two connections belong to the same internal host.

Distribution strategies. The straightforward approach to distribute the rewriting of external addresses is to first fill the flow table of the edge switch, then offload the rewriting one hop towards the internal source until the flow table of this switch is also full. The rewriting rules are then offloaded two hops away, etc. iTAP works with any distribution strategy. For example, it is conceivable to prioritize flows (for instance based on their source, destination or protocol) and to choose the de-obfuscating switch based on this priority.

DoS mitigation. While iTAP's concept of distributed rewriting provides good scalability at the network edge, it makes iTAP's anonymity guarantees depend on the number of flows. Specifically, the more outgoing connections, the higher the likelihood that the *external* endpoints of some connections are not protected within the whole network. An attacker has therefore an interest in flooding the network with external connections such that iTAP is forced to not protect the external addresses of the real user's traffic.

iTAP can protect itself from malicious hosts in two ways. *First*, iTAP can be used in conjunction with any SDN-based network anomaly detection system (e.g., [17, 26, 42]) to detect or prevent such attacks. *Second*, as iTAP is aware of the source of the flows, it can monitor the corresponding rewriting resources used at the edge for each host. If a host uses an unreasonable amount of ressources, iTAP can distribute the rewrite rules of that host instead of arbitrary ones. That way, only the flows of the malicious hosts would end up suffering from lower anonymity guarantees. Note that iTAP can be configured to not do this for important hosts.

5. CONTROLLING INFORMATION LEAKAGE

iTAP's hybrid obfuscation scheme (§4) comes with a tradeoff between scalability and anonymity. In particular, reusing the same ID across multiple flows would eventually enable an attacker to identify the bits used to encode the source and the destination of a packet. Specifically, if two observed flows have equal values at the source (resp. destination) ID bits, the attacker can correlate that they originate (resp. terminate) at the same host.

In this section, we describe how iTAP manages to (provably) prevent an attacker to infer the position of host IDs by: *(i)* measuring the information leaked to any attacker using the concept of unicity distance (§5.1); and *(ii)* adapt the rewriting before she can provably break the scheme (§5.2).

5.1 Unicity distance

The capacity of an attacker to infer which bits are used to encode the source and destination IDs depends on the number and destinations flows she can observe. In the ideal case, where the flow bitmasks used to encode the IDs appear like uniform random values and the payload of all IP packets is encrypted, it becomes impossible for an attacker to distinct between random bits and bits that are part of the obfuscation mask. In another case, where only two hosts communicate and the same source and destination IDs are used all the time, it is comparably easy to distinct between random bits and the bits that constitute the obfuscation mask. To overcome this dependency from the network's traffic characteristics, iTAP uses the so-called unicity distance to make sure that an attacker can not determine the obfuscation mask. As we will explain below, the controller uses the same mask only for a limited number of flows such that the non-random ID bits are not distinguishable from random bits.

Unicity distance. For a secret-key cipher, Shannon [31] defined the unicity distance U as the minimum number of intercepted ciphertext symbols needed, in principle, to determine the secret key. A computationally unbounded attacker which observes $N \geq U$ ciphertext symbols and uses the optimal estimation rule might be able to determine the secret key and thus to break the cipher. On the other hand, the secret key that an attacker derives when eavesdropping $N < U$ ciphertext symbols has a nonzero probability of error.

According to [31], the approximate number of intercepted ciphertext symbols, defined as the unicity distance, is

$$U = \frac{H(K)}{D} \tag{1}$$

where $H(K)$ is the entropy in the key space and D is the redundancy of the language.

5.2 When to adapt the encoding

We now explain how we use the unicity distance as a lower bound on the number of flows an attacker needs to eavesdrop until she might be able to determine the source and destination IDs of a flow. We say that an attacker is successful if she identified which bits are used for the source ID and for the destination ID, respectively. Therefore, the successful attacker will be able to determine the IDs of the source and destination of each flow. However, the attacker will not know the mapping between IDs and hosts.

The equivalent of a secret key in this case is the selection of bits used for the source and the destination ID. Then, the key must contain the following information:

- Which bits are part of the obfuscation mask (and thus used for the source ID or the destination ID);
- Which of these bits are used for the source ID and which of them are used for the destination ID.

For a header-bitstring of length l_h, there are $\binom{l_h}{l_{src}+l_{dst}}$ possible obfuscation masks to encode a source ID of l_{src} bits and a destination ID of l_{dst} bits. For each of these masks, there are $\binom{l_{src}+l_{dst}}{l_{src}} = \binom{l_{src}+l_{dst}}{l_{dst}}$ possible partitions between the source ID and the destination ID.

Since the bits are chosen uniformly at random, the entropy of a key that specifies which of the l_h bits to use as a source ID and as a destination ID computes to

$$H(K) = \log_2\left(\binom{l_h}{l_{src}+l_{dst}}\right) + \log_2\left(\binom{l_{src}+l_{dst}}{l_{src}}\right) \tag{2}$$

The second parameter in (1), the redundancy D, we see as the difference (in terms of entropy) between uniform random headers and the actually used bit-strings.

Out of the l_h available bits, $(l_{src}+l_{dst})$ are used to encode the IDs. Therefore, $l_h - (l_{src} + l_{dst})$ bits are set to uniform random values and the redundancy is upper bounded by

$$D \leq l_{src} + l_{dst} \tag{3}$$

By inserting (2) and (3) into (1), we end up with the following expression for the unicity distance:

$$U(l_h, l_{src}, l_{dst}) \geq \frac{\log_2\left(\binom{l_h}{l_{src}+l_{dst}}\right) + \log_2\left(\binom{l_{src}+l_{dst}}{l_{src}}\right)}{l_{src} + l_{dst}} \tag{4}$$

$U = 10$, for example, means that an unlimited attacker is theoretically able to determine the bits that represent the source and the destination IDs after observing 10 different flows towards the same destination[1].

6. DETECTING AND LOCALIZING AN ATTACKER

Besides passively analyzing network traffic, an attacker might want to inject traffic on intercepted links. In this section, we explain why the header rewriting performed by iTAP and the central controller significantly reduce the prospects of such an attacker.

Remember from §4 that rewritten traffic is forwarded based on its destination ID and from §3 that every packet that does not match one of the flow rules is sent to the controller. With respect to the injected traffic, the attacker has two options. She can either perform a replay attack by injecting traffic with the same header as one of the packets she eavesdropped previously, or she can inject traffic with different headers.

In the first case, the traffic will reach the destination because the header corresponds to a valid rewritten header that the controller has assigned. However, since the header is pseudo-random, the attacker neither knows the real source nor the destination of the replayed header.

In the second case, the attacker will be even less successful. If she injects traffic with a destination ID that was not previously used on this link, the next switch will not have a matching flow table entry and therefore send the packet to the controller. iTAP will come to the result that this packet does not belong to a flow that was properly rewritten and that the port where the injected packet arrived on is not connected to a host but to another switch. Therefore, the packet does neither belong to a known flow nor can it come from a legitimate host, which means it must have been illegally injected at the link connected to the port where the switch received it.

[1]Only flows towards the same destination are relevant because other flows use a different obfuscation mask.

7. PARTIAL DEPLOYMENT

iTAP can be used in networks consisting of a mixture of OpenFlow-switches and traditional MAC-learning switches. Rewriting packet headers is done by the OpenFlow-switches while the legacy switches simply forward packets based on their destination MAC address.

Forwarding packets through legacy switches. Before traversing a legacy switch, iTAP changes the destination MAC address to a MAC address belonging to the next OpenFlow-switch in the flow's path. To make sure that the legacy switches know where to forward packets, the OpenFlow switches periodically exchange packets with the source MAC address set to the ones belonging to the respective OpenFlow-switch. As a consequence of this, iTAP must not use the destination MAC field to encode the source and / or destination ID for flows that traverse legacy switches.

OpenFlow-switches only at the network edge. iTAP can protect traffic as long as there are at least two OpenFlow-switches in the path: The first OpenFlow-switch rewrites the header to a pseudo random value while the last one reverts this process. If there are two OpenFlow-switches in the path, these are ideally the ingress and the egress switches because then, iTAP protects the flow through the whole network. If the ingress and the egress switch of all flows are OpenFlow-enabled, i.e., there are OpenFlow-switches everywhere at the network edge, iTAP protects all traffic inside the network. In the evaluation in §9.2, we show that a small share of SDN switches is enough to protect a large share of the network.

Limitations. We acknowledge that partially deploying iTAP is problematic if there is no path at L2 between all hosts in a network (i.e., because there are different VLANs). Further, under certain (very rare) circumstances, the randomly chosen MAC addresses that iTAP uses to send traffic through legacy switches might collide with MAC addresses of hosts directly connected to these legacy switches. However, note that this can only happen if *(i)*, the random MAC address corresponds to the one of such a host; *(ii)*, the host is directly connected to a legacy switch that traffic with the random MAC address crosses; and *(iii)* the iTAP controller is not aware of this host (e.g., because it was not active before).

8. IMPLEMENTATION

We implemented a prototype of iTAP on top of Floodlight, a Java-based OpenFlow-controller [43]. Overall, our implementation consists of ~2,000 lines of Java-code.

Architecture. We use Floodlight and extend it by a set of modules that manage the rewriting of flow headers, the information leakage to a potential attacker and the state of switches and links.

Handling new flows. Regarding rewriting, our controller is reactive in a sense that it computes and installs rules after the first packet of a new flow enters the network. By default, each packet that does not match an existing flow rule is sent to the controller. This does also happen for the first packet of a new flow and the controller will then perform the rewriting and install the required flow rules as we described in §4.

Acting as proxy. Before the first packet of the actual flow is sent, there might be an ARP request for the destination MAC. Since the payload of ARP packets contains information that we want to hide (namely IP and MAC addresses that are used in the network), these messages cannot be flooded through the network as it would normally be done. Instead, iTAP directly answers to ARP requests. If the controller does not know the answer yet, it sends an ARP request to all hosts. These requests are sent to the switches via the secure channel and thus only cross links between hosts and switches but no links that interconnect switches.

iTAP can easily be extended to proxy messages from other protocols (such as the certificate exchange in the SSL/TLS handshake or DHCP information).

9. EVALUATION

In this section, we evaluate iTAP with respect to scalability and the protection in partial deployment settings.

9.1 Scalability

We evaluate the real-world feasibility of iTAP based on network traffic of real users. In the following, we first introduce the corresponding dataset and our processing. Then we evaluate the data plane and the control plane scalability.

9.1.1 Dataset and methodology

Baseline information. Our dataset covers the network traffic going over a core switch in our university campus network. This switch connects our research institute to the rest of the university campus network, as well as to the Internet. The switch has been configured to export connections in NetFlow v9 [2] format. We process the NetFlow data using the nfdump tools[2] and feed the data to our iTAP emulator. NetFlow exports unidirectional *flows* (i.e., a bidirectional TCP connection is reported as two flows). This makes it well suitable for our evaluation, as our SDN controller operates on the level of unidirectional flows too.

Our data spans one week (Wed–Tue) in January 2016. We observe around 100 internal IPv4 addresses and about the same number of IPv6 addresses during work days. The network is configured with a dual stack configuration, meaning that many devices will probably have an IPv4 and an IPv6 address assigned. Therefore, we conservatively estimate that our data represent the activity of around 100 devices. As our primary goal is to evaluate how many hosts our approach can handle, we exclude the activities of three devices that run an automated bitcoin experiment, causing millions of flows per day, from the presented statistics. Still, we will briefly discuss the impact of these devices and show that our approach can also handle these special traffic characteristics in §9.1.2. Table 2 summarizes our data.

Time span	# local clients/day IPv4	# local clients/day IPv6	# IPs/day	# flows/day
work days:				
5 x 24 h	97±3.8	106±6.4	56±14 k	5.2±0.7 M
weekend:				
2 x 24 h	60±2.8	46±2.8	23±2.6 k	3.1±0.05 M

Table 2: Baseline information on our dataset. The local clients displayed in the table cause 32 M flows in total. The three excluded clients running bitcoin experiments (not considered in the table) cause additional 93 M flows.

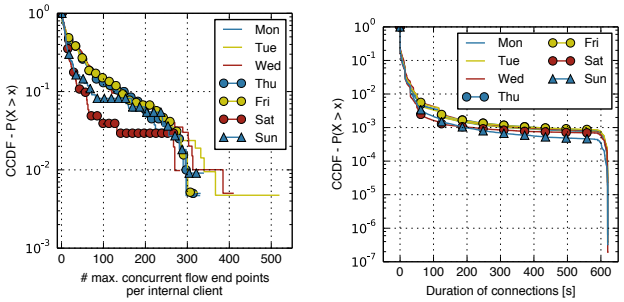
The most critical parameter affecting scalability is the number of concurrent uniflows that each client triggers, because one rule needs to be installed in an edge switch for every source and destination address (short: (src,dst)) pair. We show an upper bound for the maximum number of (src,dst) pairs per internal IP address and day in Fig. 5a. Over the whole week, only 10 % of the clients are involved in more than 170 concurrent (src,dst) pairs. The 99th percentile is ≈370, and the maximum number of (src,dst) pairs per client is ≈520.

Further, the duration of connections is relevant regarding the extrapolation presented below. The duration of most connections is small (see Fig. 5b): 95 % of all connections are not longer than 15 s and 99 % are shorter than 50 s.

Handling of incorrectly timestamped flows. Unfortunately, our dataset is affected by a problem in the NetFlow exporting procedure. The problem, which periodically occurs approximately every 68 minutes, affects the timestamps of exported flows for a duration in the order of 10 to 20 minutes. During these time intervals, the flow timestamps are shifted in time, resulting in intervals without reported flows. We cope with this issue in our simulation by only including the 30 minutes of every period in which the counts are highest. We point out that some of the incorrectly timestamped flows will appear in our analyzed time window and we will thus (incorrectly) include them in our analysis. However, including additional flows will negatively impact the measured performance. That is, we rather underestimate the performance of iTAP.

Scaling the dataset. Our NetFlow data represents the activity of around 100 internal devices. In order to evaluate the scalability of our approach for more clients, we introduce additional users by copying the behavior patterns of the real

[2]https://sourceforge.net/projects/nfdump/



(a) Max. # concurrent src/dst address pairs per internal client (upper bound).

(b) Duration of connections.

Figure 5: Connection statistics for each of the seven days.

users, i.e., we create *clones* of the clients. We clone a client by selecting all its flows and injecting the flows again with *(i)* a different client IP address at *(ii)* a randomly selected later point in time. We place each cloned flow at least one minute apart from the original flow and any previous clone of the same flow. Because 99 % of the flows are shorter than 50 s (see Fig. 5b), this means that the activities of the original and the cloned client(s) will barely overlap[3]. In total, we create three clones of every internal client IP, resulting in a dataset that is scaled by a factor of four containing 128 M flows representing the activity of 400 internal devices.

Protecting the privacy of users during our analysis. As our evaluation is based on real user data, we take several measures to protect the privacy of the users. In particular, we process the NetFlow data on an isolated, locked-down server, which only project workers can access. The project workers are well-aware of the sensitive nature of the data and conduct the experiments according to a code of ethics. We anonymize the internal network's IP addresses during processing and do not report any IP addresses in our work. All data are stored on encrypted hard disks. Further, the published information has been verified and declassified by the data provider.

9.1.2 Data plane scalability

Regarding scalability in the data plane, the most crucial factor is the number of flow rules installed in switches. In the following paragraphs, we show that iTAP can operate within the typical resource constraints of commodity hardware on core and edge switches. Fig. 6 summarizes our evaluation.

Flow rules in core switches. In the core switches, packets are forwarded based on their destination ID. This means, a core switch needs to contain one forwarding rule per destination ID or, in other words, one rule per host that communicates via this switch. Our evaluation based on 400 users and one week of traffic shows that the total number of flow rules does not exceed 2.5k (see Fig. 6, top left). This is perfectly manageable for the current generation of OpenFlow switches which can handle thousands [6] up to millions [1] of flow table entries.

Flow rules in edge switches. Edge switches take care of (de-)anonymizing flows that (exit) enter the network. For example, to deanonymize a packet before it is transmitted to an attached host, the switch replaces the randomized source and destination addresses with the original MAC and IP addresses. To this end, the edge switch requires one flow rule for every (source, destination) pair. That is, the first TCP connection a client C established with a server S results in two pairs (C, S) and (S, C), and therefore two new rules. Additional concurrent connections between C and S do *not* require any additional flow rules.

We distinguish between two kinds of edge switches: *(i)* internal commodity edge switches to which clients are attached (e.g., in an office room) and *(ii)* the edge switch/router connecting the enterprise network to the Internet.

For **case *(i)***, the clients are physically connected to these switches via a LAN cable. We don't have detailed information on how clients are connected to edge switches in the evaluated network. Therefore, we present in the following a kind of worst case scenario: Typical larger commodity SDN switches have 24 or 48 LAN ports and can handle 10 k flow rules. As previously discussed, nine out of ten clients (90th percentile) cause less than 170 flow entries during peak times. Assuming that 48 clients have their daily peaks at the same time and all of these clients concurrently generate 170 flow entries We still only need 8.2 k (48 $\times$ 170) rules, which is well within the maximum number of installable rules.

For **case *(ii)***, the worst case is that all uniflows leave the network or come from externally, i.e., they all need to pass the edge switch/router. The corresponding simulation results are shown in Fig. 6, top right. The maximum is around 30 k.

Also without excluding the hosts running the bitcoin experiments, the maximum number of flow table entries would still be in a feasible range (around 8 k for the original dataset with 100 users). But, the number of flow rules at the edge switch connected to these three hosts would exceed 10 k.

To cope with the problem that the Internet/WAN-facing edge switches might become overloaded, we introduced a distributed rewriting technique in §4.2. Performing this technique at the Internet/WAN-facing network edge, iTAP avoids exceeding the flow table capacities of the switch.

In Fig. 9, we show the effect of offloading the rewriting of certain external addresses from the Internet-facing edge switch to a core switch. The plot shows the situation at one point in time (the second with the highest number of flow rules at the edge switch) for our original dataset containing traffic of 100 users. We simulated the straightforward distribution strategy of offloading destinations according to their popularity (i.e., the destination accounting for most flows is distributed first). We motivate this strategy by the assumption that the more popular an external address is, the less it is a secret that hosts connect to it. However, as we have explained in §4.2, the network operator is free to choose any other distribution strategy. As Fig. 9 shows,

[3]The larger the offset, the more flows can potentially be placed in the regions affected by the timestamp bug. This is why we do not use a larger time offset.

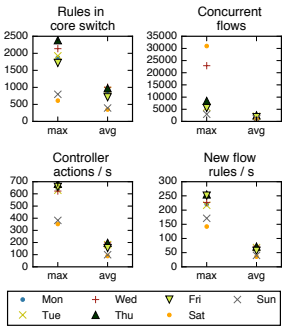


Figure 6: iTAP scales to large networks: These graphs are based on seven days of traffic data representing 400 hosts.

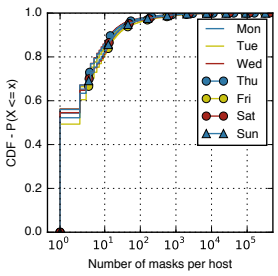


Figure 7: iTAP overhead is small: For 50 % of the hosts, it needs to compute only one destination ID and mask per day.

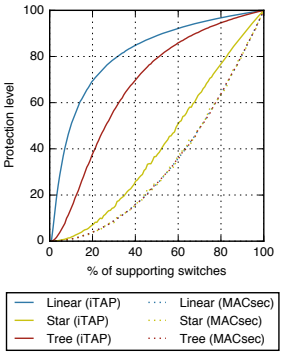


Figure 8: Only a small share of SDN-enabled switches is sufficient for iTAP to protect a large share of the network traffic.

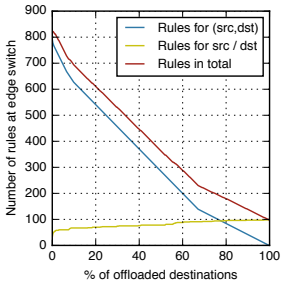


Figure 9: Offloading a small share of the external destinations results in a significant decrease in the number of required flow rules.

offloading only a few destinations results in a significant decrease in the required number of flow rules. Thus, by distributing the rewriting of *external* addresses, iTAP scales even at the Internet/WAN-facing network edge while still providing anonymity for all internal addresses.

9.1.3 Control plane scalability

We now show that iTAP is able to handle the load imposed by the network. iTAP's controller needs to *(i)* act on events trigged by switches when new flows arrive and *(ii)* install corresponding obfuscation rules on switches. In this section, we evaluate the number of flows that the controller needs to handle, measure the number of newly installed rules, and compare these numbers to a standard SDN deployment.

Controller actions. If a packet does not match an existing flow rule, the switch sends a "packet-in" message to the controller. Edge switches trigger a packet-in message if there is no rule for the (source, destination) tuple of the newly arrived packet. In our evaluation, we observed a maximum of 700 packet-in messages per second (see Fig. 6, bottom left). Such a load is perfectly reasonable as current controller platforms can handle millions of packet-ins per second [38].

Flow table updates. Upon an incoming packet-in message, the controller pushes new rules to the switches via flow-mod messages. At the edge switches, each packet-in message will be followed by a new flow rule, while in the core switches, a new rule is only required if the destination is not used yet. As shown in Fig. 6 (bottom right), there were at most 250 new rules to install per second in the core switch while a maximum of 700 new flows (Fig. 6, bottom left) per second were reported to the controller. Again, such a number of table updates is well within the capabilities of existing SDN switches for which installing a new rule is done within 1 millisecond, on average [29].

Anonymity overhead. We now evaluate iTAP overhead with respect to a normal SDN controller which only needs to act once per destination address that appears in the network. In Fig. 7, we show that this also holds for around 50 percent of the hosts when using iTAP. For these hosts, the controller can use the same destination ID and mask over the whole day without violating our protection guarantees (§5). For the other hosts, the controller needs to switch the obfuscation masks during a day, but as we showed above, the controller can easily handle this load.

9.2 Partial deployment

While iTAP can be deployed if only a small share of the switches support SDN (§7), this results in weaker anonymity guarantees. In the following, we evaluate the amount of protection iTAP provides in different partial deployment scenarios. As baseline, we consider the protection level provided by a corresponding, partially deployed MACsec network.

We do the analysis for a linear, star and binary tree topology, each consisting of 127 switches. We gradually increase the percentage of OpenFlow switches from 0 % to 100 % and place the OpenFlow-capable switches at random positions in the network. Fig. 8 shows the percentage of OpenFlow-capable switches on the x-axis, and the resulting protection level (averaged over 1000 runs of the simulation).

We compute the protection by simulating a flow between every pair of edge switches (the leaf nodes of the star and tree topology, and all nodes of the linear topology) and compute the percentage of links that are protected. The total protection level plotted in Fig. 8 is the average protection value across all connections.

While iTAP protects traffic between the first and the last OpenFlow switch in the path, MACsec can only protect traffic on enabled links. This explains the significantly lower protection level for partial deployment.

10. DISCUSSION

In this section, we briefly discuss different failure and attack scenarios.

Attackers on first/last hops. While iTAP fully protects internal links (see Fig. 2), it does not protect host-facing links. If an attacker has intercepted a link between a host and an edge switch, she can observe unprotected traffic from and to this host. But this does not allow an attacker to draw any conclusions about the traffic of other, unrelated hosts nor does it help to break the anonymity of other flows. That is, the impact of the attacker is limited to the eavesdropped host's communication. As an additional protection measure, one can install a virtual SDN switch (e.g., [44]) on the host. This makes the "first hop" internal to the host. Such a setup provides *communication anonymity* of remote hosts, but it does not prevent an attacker from analyzing the traffic patterns of the tapped host.

Compromised switches. If an attacker has access to a switch, for example through a backdoor in the switch's firmware, she can read and modify the flow table. In a core switch, the flow table reveals the destination IDs used on this switch and the corresponding actions (output at which port). But the flow table neither reveals the used source IDs nor the real source and destination addresses of packets.

In the flow tables of edge switches, the attacker can see the mapping from the real header h to the rewritten header $\tilde{h}$ for all packets from or to hosts connected to the switch. An attacker that can alter the flow table of a switch can perform a DoS attack by dropping packets. Further, an attacker can send spoofed messages to the controller. Such attacks are outside of our scope but addressed in existing work [23].

Compromised controller. As any SDN-based solution, iTAP security inherently depends on the security of the controller platform. Several works looked at the problem of securing SDN controllers, especially protecting them from rogue applications or users [23, 27, 28]. A recent survey is available in [8]. Since iTAP relies on few SDN primitives, its logic could easily be implemented in any of these platforms.

Controller failure. If the controller crashes and loses its state, this does not affect existing connections because for these, the required flow rules are already installed. As soon as the controller is up again, it will build its new state from the newly arriving packets. For these new connections, it will choose new source and destination IDs which means that the protection guarantees do still hold. If the controller is not available, switches will not receive commands how to handle new connections and therefore drop the packets.

Malicious overloading of switches. An attacker on an end host might intentionally cause a large number of flows to different hosts to exhaust the switches' flow rule tables. As iTAP's controller has a global view over all connections in the network, it can easily prevent this attack by limiting the number of obfuscated flows (and thus flow rules) that a host can initiate. If an end point exceeds the limit, new flows could either be blocked or be forwarded unobfuscated.

11. RELATED WORK

iTAP is related to existing work in network obfuscation, moving target defense and anonymous communication.

Network obfuscation and moving target defense. The topic of dynamically changing network configurations in order to present a misleading view to an attacker has been addressed in a variety of proposals in literature. Previous work describes concepts to obfuscate traffic by changing the network architecture (e.g., [11]), by applying cryptographic algorithms on packets [21], by randomly changing the IP addresses and TCP/UDP ports [9,20,25,32,39,40], by means of multipath routing [16] or by protocol-specific modifications of packets [19]. However, existing solutions require changes at hosts [21, 32, 39, 40] and/or are only usable for specific operating systems, applications or protocols [16, 19, 25, 32]. Solutions that can be deployed without modifying hosts require additional middleboxes [11, 21], tamper with existing connections [9] or suffer from synchronization problems [21, 39]. A recent publication describes PHEAR [33], a system that replaces source and destination addresses by random values (nonces). This concept is similar to iTAP but while it additionally encrypts packet data beyond the network layer, it does not address scalability issues (number of flow rules) and the rewriting / encryption is handled by a proxy that needs to be installed at hosts.

Anonymous communication and encryption. Onion Routing [18] and its implementation in TOR [14] are probably the best known solutions for providing anonymous communications. Recently, a couple of systems have been proposed that bear similarities with Tor but provide better scalability, performance or traffic analysis prevention [12, 13, 15, 24]. In contrast to iTAP, these systems require changes at the end hosts or a whole new network architecture.

Encryption schemes such as IPsec [45] and MACsec [5] also provide anonymous communication to a certain extent but they do not hide the endpoints connections (IP / MAC addresses). Further, these solutions require support from the endpoints (IPsec) or all switches in the network (MACsec).

12. CONCLUSION

In this paper, we presented iTAP, a novel anonymity framework, which enables anonymous communication *within* the premises of a network. iTAP is an in-network system, it requires no changes to hosts and can be partially deployed.

iTAP achieves its anonymity properties by rewriting packet headers at the network edge. While the idea of rewriting packet headers is simple, doing it at scale while guaranteeing strong anonymity is challenging. We solve this challenge by introducing a novel rewriting scheme which combines the anonymity benefits of having a single-use ID per flow with the scalability benefits of having a constant ID per host.

We implemented iTAP on top of an existing SDN controller and evaluated its usability and performance using real network traffic traces. We show that iTAP works in practice, on existing hardware. Moreover, a few SDN switches are enough to protect a large share of the network traffic.

13. REFERENCES

[1] B1SDN product brief. http://www.znyx.com/wp-content/uploads/2015/05/B1_SDN_brief_101414_web.pdf.

[2] Cisco IOS NetFlow. http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html.

[3] Enterprise campus 3.0 architecture: Overview and framework. http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Campus/campover.html.

[4] IBM x-force threat intelligence quarterly, 2Q 2015. https://public.dhe.ibm.com/common/ssi/ecm/wg/en/wgl03076usen/WGL03076USEN.PDF.

[5] Media access control (MAC) security. http://standards.ieee.org/getieee802/download/802.1AE-2006.pdf.

[6] Noviswitch 2122 high performance openflow switch. http://noviflow.com/wp-content/uploads/NoviSwitch-2122-Datasheet-V2_1.pdf.

[7] RFC 6325 - routing bridges (rbridges). https://tools.ietf.org/html/rfc3031, July 2011.

[8] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov. Security in software defined networks: a survey. *Communications Surveys & Tutorials, IEEE*, 17(4), 2015.

[9] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis. Defending against hitlist worms using network address space randomization. *Computer Networks*, 51(12), 2007.

[10] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al. P4: Programming protocol-independent packet processors. *ACM CCR*, 44(3), 2014.

[11] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker. Sane: A protection architecture for enterprise networks. In *USENIX Security*, 2006.

[12] D. Chaum, F. Javani, A. Kate, A. Krasnova, J. de Ruiter, and A. T. Sherman. cmix: Anonymization by high-performance scalable mixing.

[13] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. Hornet: High-speed onion routing at the network layer. In *ACM SIGSAC*, 2015.

[14] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC, 2004.

[15] K. P. Dyer, S. E. Coull, and T. Shrimpton. Marionette: A programmable network traffic obfuscation system. In *USENIX Security*, 2015.

[16] E. Germano da Silva, L. A. Dias Knob, J. A. Wickboldt, L. P. Gaspary, L. Z. Granville, and A. Schaeffer-Filho. Capitalizing on SDN-based SCADA systems: An anti-eavesdropping case-study. In *IFIP/IEEE IM*, 2015.

[17] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks*, 62, 2014.

[18] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2), 1999.

[19] K. E. Huber. Host-based systemic network obfuscation system for windows. Technical report, DTIC, 2011.

[20] J. H. Jafarian, E. Al-Shaer, and Q. Duan. Openflow random host mutation: transparent moving target defense using software defined networking. In *ACM HotSDN*, 2012.

[21] D. Kewley, R. Fink, J. Lowry, and M. Dean. Dynamic approaches to thwart adversary intelligence gathering. In *IEEE DARPA DISCEX*, volume 1, 2001.

[22] O. Khazan. The creepy, long-standing practice of undersea cable tapping. http://www.theatlantic.com/international/archive/2013/07/id/277855/, July 2013.

[23] D. Kreutz, F. Ramos, and P. Verissimo. Towards secure and dependable software-defined networks. In *ACM HotSDN*, 2013.

[24] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM CCR*, volume 43, 2013.

[25] D. C. MacFarland and C. A. Shue. The SDN shuffle: Creating a moving-target defense using host-based software-defined networking. In *ACM MTD*, 2015.

[26] S. A. Mehdi, J. Khalid, and S. A. Khayam. Revisiting traffic anomaly detection using software defined networking. In *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2011.

[27] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In *ACM HotSDN*, 2012.

[28] P. A. Porras, S. Cheung, M. W. Fong, K. Skinner, and V. Yegneswaran. Securing the software defined network control layer. In *NDSS*, 2015.

[29] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore. Oflops: An open framework for openflow switch evaluation. PAM, Berlin, Heidelberg, 2012. Springer-Verlag.

[30] M. Seaman. Shortest path bridging. http://ieee802.org/1/files/public/docs2005/new-seaman-shortestpath-par-0405-02.htm, 2005.

[31] C. E. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4), 1949.

[32] L. Shu and W. Weinstein. Camouflage of network traffic to resist attack, Jan. 30 2007. US Patent 7,171,493.

[33] R. Skowyra, K. Bauer, V. Dedhia, and H. Okhravi. Have no phear: Networks without identifiers. In *ACM MTD*, 2016.

[34] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter. Past: Scalable ethernet for data centers. In *ACM CoNEXT*, 2012.

[35] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal. RAPTOR: Routing attacks on privacy in TOR. In *USENIX Security*, 2015.

[36] C. Timberg. Google encrypts data amid backlash against NSA spying. http://wapo.st/1adFyAe.

[37] L. Vanbever, O. Li, J. Rexford, and P. Mittal. Anonymity on quicksand: Using BGP to compromise TOR. In *ACM HotNets*, 2014.

[38] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak. Maple: simplifying SDN programming using algorithmic policies. In *ACM CCR*, volume 43, 2013.

[39] F. Webber, P. P. Pal, M. Atighetchi, C. Jones, and P. Rubel. Applications that participate in their own defense (apod). Technical report, DTIC, 2003.

[40] W. Weinstein and J. Lepanto. Camouflage of network traffic to resist attack (contra). In *DARPA DISCEX*, volume 2, 2003.

[41] N. Zaidenberg and A. Resh. Timing and side channel attacks. In *Cyber Security: Analytics, Technology and Automation*. Springer, 2015.

[42] Y. Zhang. An adaptive flow counting method for anomaly detection in SDN. In *ACM CoNEXT*, 2013.

[43] Floodlight openflow controller. https://github.com/floodlight/floodlight.

[44] Open vswitch. http://openvswitch.org/.

[45] RFC 4301 - security architecture for the internet protocol. https://tools.ietf.org/html/rfc4301, Dec 2005.